WEST Search History

DATE: Saturday, June 22, 2002

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=ADJ			
L1	(extension near6 command) same script same pars$	2	L1

END OF SEARCH HISTORY

WEST

End of Result Set

Generate Collection Print

L3: Entry 11 of 11 File: USPT Jun 29, 1999

DOCUMENT-IDENTIFIER: US 5916310 A
TITLE: Method and system for enhancing keyboard functionality in an HTML document

Brief Summary Paragraph Right (10):
Mechanisms exist for enhancing HTML/browser functionality. Microsoft Corp. offers component technology referred to as "ActiveX" that operates as a "plug-in" to the MICROSOFT EXPLORER Web browser. ActiveX however, is a proprietary technology and currently does not execute in the most popular Web browser, NAVIGATOR, offered by Netscape Communications. Moreover, ActiveX extensions are widely regarded as a security exposure due to the way in which they interact with the client computer's operating system.

Brief Summary Paragraph Right (11):
The JavaScript scripting language is another mechanism for enhancing the functionality of an HTML page. JavaScript instructions can be embedded in an HTML document to perform a variety of different tasks. However, JavaScript does not have access to keyboard keystrokes, and thus cannot be used by itself to provide enhanced keyboard functionality. Java applets are another mechanism by which an HTML document can be enhanced. An applet is an executable code file that is downloaded and interpreted by the browser along with the HTML document. An applet "owns" only a particular subset of the HTML page, and has no control or interaction with page locations that it does not "own."

Brief Summary Paragraph Right (17):
To achieve the foregoing and other objects and in accordance with the purposes of the present invention as described above, a method and system for enhancing keyboard functionality in an HTML document is provided. An HTML document is generated which includes a script application and an applet, the script application communicating with the applet over a bi-directional programming communications interface. The HTML document is downloaded from a server to a browser executing on a client computer. Upon a change in focus on the HTML document, an event handler in the script application is invoked. The event handler invokes a method in the applet requesting that the applet reacquire the focus. Each key press is intercepted by the applet. The applet then invokes a key press action function in the script application which provides predetermined functionality for the respective key press. Such interaction between the script application and the applet achieves complete control over each key press on the keyboard and enables implementation of any desired functionality associated with any key on a conventional computer keyboard.

Brief Summary Paragraph Right (18):
According to one embodiment of this invention, the applet categorizes each key into one of three key categories, a data key category, a data edit key category, or a function key category. The applet then invokes a particular function in the script application depending upon the category of the key. The script application is passed a code identifying the key which was pressed and implements the desired functionality.

Brief Summary Paragraph Right (19):
If a data key is pressed, the script application can determine the text field in which the data key is to be inserted, and can update the existing string in such data field either by inserting the data key character at the current location of the cursor, or by replacing a character with the data key character depending upon whether insert mode is on or off. If the key pressed is a data edit key, the script application can implement the predetermined edit sequence as a function of the particular data edit key. If the key pressed is a keyboard function key, the script application can simulate the

activation of a respective function key via its corresponding on-screen button.

Brief Summary Paragraph Right (20):
The script application simulates a "focus" by displaying a blinking cursor in the user-selected text field. The script application also maintains the current cursor location. Such cursor location information can be provided to an application to utilize as desired.

Detailed Description Paragraph Right (2):
Referring now to FIG. 1, a schematic diagram including a browser 20, computer screen 22 and keyboard 24 is shown. Browser 20 receives an HTML page 23 containing "tags" (HTML instructions), Universal Resource Locators (URL) to other files, and other HTML constructs from an application 30 over a network 29. Browser 20 includes the intelligence to interpret this information and generate and display HTML page 23 on computer screen 22. Arrow 21 illustrates that browser 20 interacts, both on output and input, with HTML page 23 displayed on computer screen 22. HTML page 23 typically contains various elements, including "form" elements such as text fields 26 and 28, as well as other user interface components such as buttons 35-37. Such HTML user interface (UI) components (or "widgets") are well known to those skilled in the art. A user can use a conventional keyboard 24 to enter data into a text field of HTML page 23, such as text field 28.

Detailed Description Paragraph Right (7):
Application 50 and applet 48 communicate with each other over interface 52. Interface 52 can comprise a LiveConnect interface available in various web browsers, including the NETSCAPE NAVIGATOR browser. Other such communication interfaces which achieve such interaction between application 50 and applet 48 would also be suitable for use in the present invention. Development of JavaScript applications is described in many books, magazines and manuals, such as, for example Teach Yourself JavaScript 1.1 in One Week, Second Edition, Denesch. Similarly, developing Java applets is known to those skilled in the art, and is taught in many Java programming books. The programming instructions which compose a JavaScript application are typically embedded directly into HTML page 25. The programming instructions for Java applet 48 are provided in a separately precompiled code-file referred to as a ".Class" file, and is referenced by a URL in HTML page 25.

CLAIMS:

1. A method for enhancing keyboard functionality in an HTML page, comprising:

providing a script application and an applet in an HTML page;

communicating the HTML page to a browser; providing an interface between the script application and the applet;

implementing an event handler in one of the script application and the applet operative to be invoked upon a change in focus on the HTML page;

implementing a focus method in the applet operative to obtain a keyboard focus upon request by the event handler; and

implementing a keypress action function in the script application operative to simulate a respective keypress upon notification by the applet.

3. A method according to claim 2, further comprising providing a plurality of keypress action functions in the script application.

4. A method according to claim 3, further comprising receiving notification of the keypress, categorizing the keypress as one of a plurality of different types of keys, and invoking one of the plurality of keypress action functions in the script application as a function of the category of the keypress.

9. A method according to claim 2, wherein the script application and the applet communicate with one another over a LiveConnect interface.

10. A system for extending keyboard functionality in an HTML page, comprising:

an HTML page containing therein a script application and an applet;

a browser operative to interpret and execute the script application and the applet;

an interface to enable communication between the script application and the applet;

the applet including a request focus method operative to respond to a request from the script application to obtain a keyboard focus; and

the script application having a keypress action method operative to simulate the keypress upon request by the applet.

12. A system according to claim 10, wherein the applet is operative to receive notification of the keypress, and invoke the keypress action function in the script application.

13. A system according to claim 10, wherein the script application further comprises a plurality of keypress action functions, and wherein the applet is operative to receive notification of the keypress, categorize the keypress as one of a plurality of different types of keys, and invoke one of the plurality of keypress action functions in the script application as a function of the category of the keypress.

14. A method for enhancing keyboard functionality in an HTML page, comprising:

communicating an HTML page having a script application and an applet to a browser;

receiving, by the script application, a change in focus on the HTML page and invoking the applet to maintain the focus;

receiving, by the applet, information identifying a key pressed by a user;

invoking a method in the script application to provide a predetermined functionality associated with the key.

15. A method according to claim 14, further comprising categorizing, by the applet, each key into one of a plurality of key categories, and wherein the invoking a method step comprises invoking a respective method in the script application as a function of the category of the key.

16. A method according to claim 10, wherein the applet and the script application communicate with each other via a bidirectional communications interface.

17. A method for enhancing keyboard functionality in an HTML page, comprising:

providing an HTML page with an applet;

communicating the HTML page to a browser which includes a script language interpreter and provides an interface between the applet and the script language interpreter;

maintaining the keyboard focus with the applet;

obtaining in the applet key information identifying a key pressed on a keyboard; and

interacting with the script language interpreter to provide predetermined functionality associated with the key.

19. A method according to claim 17, wherein the interacting with the script language interpreter comprises invoking a system function.

20. A method according to claim 17, wherein the interacting with the script language interpreter comprises invoking a user-written function which is interpreted by the script language interpreter.

WEST

Generate Collection Print

L3: Entry 10 of 11 File: USPT Mar 6, 2001

DOCUMENT-IDENTIFIER: US 6199046 B1
TITLE: Method system and article of manufacture for performing real time currency conversion

Abstract Paragraph Left (1):
A currency converter consisting of a source of exchange rate information with either an object performing the role of currency selection, exchange rate information retrieval, and price display, or objects undertaking currency selection and exchange rate information retrieval and price display. Prices embedded within the document or environment are displayed in the default or selected currency. The appropriate currency for price display is selected by using methods such as a menu used by the user, or by reading system information to choose the currency. The exchange rate information is retrieved from the source of exchange rate information and is then passed to any other price display object or objects, and optionally, any other currency selection and exchange rate retrieval objects or currency selection, exchange rate information retrieval, and price display objects in the document or environment.

Brief Summary Paragraph Right (23):
The prior art has failed to realize that the price observed in a networked environment can be a dynamic object which relies on information retrieved over a communications network which is distinct from the information in which it is embedded.

Detailed Description Paragraph Right (45):
With Java, developers can create robust User Interface (UI) components. Custom "widgets" (e.g. real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

Detailed Description Paragraph Right (47):
Another technology that provides similar function to JAVA is provided by Microsoft and ActiveX Technologies, to give developers and Web designers wherewithal to build dynamic content for the Internet and personal computers. ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over 100 companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programmming languages including Microsoft Visual C++, Borland Delphi, Microsoft Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications.

Detailed Description Paragraph Right (54):
An alternative implementation of the embodiment depicted in FIG. 6 is depicted in FIG. 7, whereby each of the currency selection, exchange rate retrieval and price display objects (2a)-(2n) passes information from one object to the next in a "relay" fashion. This implementation is preferable if the information must be passed from one currency selection, exchange rate retrieval and price display object to another, and the currency selection, exchange rate retrieval and price display objects are embedded in a sequence of documents or environments enabling the order of transfer of information to be efficiently deduced. The currency selection, exchange rate retrieval and price display object (2a) that accesses the exchange rate information source (1) would typically, but not necessarily, be the currency selection, exchange rate retrieval and

price display object with which the user interacts to change the currency.

Detailed Description Paragraph Right (76):
A preferred embodiment consists of any one or combination of the following objects or groups of objects utilized to display prices embedded within a document or environment:

Detailed Description Paragraph Right (82):
The source from which the exchange rate information is transferred need not necessarily be the source from which the documents or environments using the currency selection, exchange rate retrieval and price display objects (2), the currency selection and exchange rate retrieval object (3) or price display object (4) are retrieved. For example, a document or environment on the World Wide Web may embed a reference to a price display object that is located at a different source altogether, which utilizes exchange rate information at a different source again.

Detailed Description Paragraph Right (93):
The currency selection, exchange rate retrieval and price display object (2) can be used to embed a price or prices within the text, images or objects of a document or environment such as, but not limited to, a Hypertext Markup Language (HTML) document, a Virtual Reality Modeling Language (VRML) environment, or a Multi User Dimension (MUD).

Detailed Description Paragraph Right (127):
In a preferred embodiment, the parameters embedded within the document or environment that the currency selection, exchange rate retrieval and price display object (2) views would be formatted such that the default price to be displayed with its default currency symbol would still be displayed by earlier software, such as but not limited to, older World Wide Web (WWW) browsers, which are unable to recognize the tag representing a price field.

Detailed Description Paragraph Right (128):
Alternatively, the software, such as but not limited to a World Wide Web (WWW) browser, could implement the currency converter by using a "plug-in" which would perform the function of the currency selection, exchange rate retrieval and price display object (2) within the software.

Detailed Description Paragraph Right (133):
Someone skilled in the art of computer programming can duplicate the functionality of the currency converter in another programming language, software environment, software package, network environment, or another medium in which prices are embedded.

Detailed Description Paragraph Right (134):
A skilled programmer could also implement the currency converter as a readily embedded object able to be used by others within their programs, operating systems or environments.

Detailed Description Paragraph Right (135):
One of ordinary skill in the art of computer programming and Hypertext Markup Language design could implement the functionality of the currency converter using scripts or macro languages embedded within Hypertext Markup Language documents or, more generally, in any document or environment whose viewing software supports the execution of embedded macros or scripts. For example, the currency converter could be implemented within Dynamic Hypertext Markup Language (Dynamic HTML) to display prices in the user's selected currency. Scripts or macros could also interact with embedded software objects such as, but not limited to, applets or ActiveX controls to implement the functionality of the currency converter.

Detailed Description Paragraph Type 1 (19):
one or a plurality of other currency selection and exchange rate retrieval objects embedded within the document, environment, or operating system (3b)-(3n).

Detailed Description Paragraph Type 1 (22):
one or a plurality of other currency selection and exchange rate retrieval objects embedded within the document, environment, or operating system (3a)-(3n).

Detailed Description Paragraph Type 1 (38):
A method which accepts the parameters of the default currency, the price or prices to be displayed in the default currency, and in a preferred implementation, accepts cosmetic parameters which allow the price to be displayed seamlessly within the

document or environment, such as, but not limited to, the color of the price, the font size of the price, and the price text's background color. The form of the parameters can be, but is not limited to, parameters embedded within a Hypertext Markup Language (HTML) document, or parameters in a file separate from the document, environment, or operating system in which the prices are displayed. The document or environment in which the prices are displayed can be dynamically generated or exist in a static form.

Detailed Description Paragraph Type 1 (46):
A method which accepts the parameters of the default currency. The form of the parameters can be, but is not limited to, parameters embedded within a Hypertext Markup Language (HTML) document, or parameters in a file separate from the document or environment in which the prices are displayed.

Detailed Description Paragraph Type 1 (52):
A method which accepts the parameters of the default currency, the price or prices to be displayed in the default currency, and in a preferred implementation, accepts cosmetic parameters which allow the price to be displayed seamlessly within the document, environment, or operating system, such as the color of the price, the font size of the price, and the price text's background color. The form of the parameters can be, but is not limited to, parameters embedded within a Hypertext Markup Language (HTML) document, or parameters in a file separate from the document or environment in which the prices are displayed.

Detailed Description Paragraph Type 1 (53):
A method by which the object displays the price or prices in the default or selected currency within the document or environment such as, but not limited to, passing a text string to a node within a Virtual Reality Modeling Language (VRML) environment, or providing an image of the price to be displayed for embedding within a Hypertext Markup Language (HTML) document, or displaying the entire document or environment with all prices in the default or selected currency

Detailed Description Paragraph Type 1 (79):
the displayed price can consist of a picture embedded within the document or environment

Detailed Description Paragraph Type 1 (80):
the displayed price can consist of a text string embedded within the text of the document or environment

Detailed Description Paragraph Type 1 (81):
the displayed price can consist of either a text string or a picture that is embedded within a picture in the document or environment

Detailed Description Paragraph Type 2 (5):
providing an image of the price to be displayed for embedding within a Hypertext Markup Language (HTML) document,

Detailed Description Paragraph Table (8):
Begin File Listing: AdsuraException.h #ifndef _ADSURAEXCEPTION_H #define _ADSURAEXCEPTION_H class CAdsuraException : public CException { DECLARE_DYNCREATE(CAdsuraException) public: CAdsuraException(int nCode = 0); .about.CAdsuraException() {} int m_nErrorCode; }; void ThrowAdsuraException(int nCode); #endif End File Listing: AdsuraException.h Begin File Listing: AdsuraException.cpp #include "stdafx.h" #include "AdsuraException.h" IMPLEMENT_DYNCREATE(CAdsuraException, CException) CAdsuraException::CAdsuraException(int nCode) :m_nErrorCode(nCode) { } void ThrowAdsuraException (int nCode) { CAdsuraException* pEx = new CAdsuraException(nCode); throw pEx; } End File Listing: AdsuraException.cpp Begin File Listing: AdsuraExchangeView.h #if !defined(AFX_ADSURAEXCHANGEVIEW_H_B8189F1A_CD0B_11D0_86E7_0000B43A75C9_INCLUDED_) #define AFX_ADSURAEXCHANGEVIEW_H_B8189F1A_CD0B_11D0_86E 7_0000B43A75C9_INCLUDED.sub.-- #if _MSC_VER >= 1000 #pragma once #endif // MSC_VER >= 1000 // AdsuraExchangeView.h : main header file for ADSURAEXCHANGEVIEW.DLL #if !defined(_AFXCTL_H_) #error include `afxctl.h` before including this file #endif #include "resource.h" // main symbols // ////////////////////////// // CAdsuraExchangeViewApp: See AdsuraExchangeView.cpp for implementation. class CAdsuraExchangeViewApp : public COleControlModule { public: BOOL InitInstance(); int ExitInstance(); }; extern const GUID CDECL _tlid extern const WORD _wVerMajor; extern const WORD _wVerMinor; //{{AFX_INSERT_LOCATION}} // Microsoft Developer Studio will insert additional declarations immediately before the previous line. #endif //

```
!defined(AFX_ADSURAEXCHANGEVIEW_H_B8189F1A_CD0B_1 1D0_86E7_0000B43A75C9_INCLUDED) End
File Listing: AdsuraExchangeView.h Begin File Listing: AdsuraExchangeView.def
:AdsuraExchangeView.def : Declares the module parameters. LIBRARY
"ADSURAEXCHANGEVIEW.OCX" EXPORTS D11CanUnloadNow @ 1 PRIVATE D11GetClassObject @ 2
PRIVATE D11RegisterServer @ 3 PRIVATE D11UnregisterServer @ 4 PRIVATE End File Listing:
AdsuraExchangeView.def Begin File Listing: AdsuraExchangeView.cpp //
AdsuraExchangeView.cpp : Implementation of CAdsuraExchangeViewApp and DLL registration.
#include "stdafx.h" #include "AdsuraExchangeView.h" #ifdef _DEBUG #define new DEBUG_NEW
#undef THIS_FILE static char THIS_FILE[] = __FILE__; #endif CAdsuraExchangeViewApp NEAR
theApp; const GUID CDECL BASED_CODE _tlid = {0xb8189f11, 0xcd0b, 0x11d0, {0x86, 0xe7,
0, 0, 0xb4, 0x3a, 0x75, 0xc9}}; const WORD _wVerMajor = 1; const WORD _wVerMinor = 0;
////////////////////////////////////////////////// ////////////////////////// //
CAdsuraExchangeViewApp::InitInstance - DLL initialization BOOL
CAdsuraExchangeViewApp::InitInstance() { BOOL bInit =
COleControlModule::InitInstance(); if (bInit) { // TODO: Add your own module
initialization code here. } return bInit; }
////////////////////////////////////////////////// ////////////////////////// //
CAdsuraExchangeViewApp::ExitInstance - DLL termination int CAdsuraExchangeViewApp:
:ExitInstance() { // TODO: Add your own module termination code here. return
COleControlModule::ExitInstance(); }
////////////////////////////////////////////////// ////////////////////////// //
D11RegisterServer - Adds entries to the system registry STDAPI D11RegisterServer(void)
{ AFX_MANAGE_STATE(_afxModuleAddrThis); if
(!AfxOleRegisterTypeLb(AfxGetInstanceHandle(), _tlid)) return ResultFromScode(SELFREG
E_TYPELIB); if (!COleObjectFactoryEx::UpdateRegistryAll(TRUE)) return
ResultFromScode(SELFREG_E_CLASS); return NOERROR; }
////////////////////////////////////////////////// ////////////////////////// //
D11UnregisterServer - Removes entries from the system registry STDAPI
D11UnregisterServer(void) { AFX_MANAGE_STATE(_afxModuleAddrThis); if
(!AfxOleUnregisterTypeLib(_tlid, _wVerMajor, _wVerMinor)) return
ResultFromScode(SELFREG_E_TYPELIB); if (!COleObjectFactoryEx::UpdateRegistryAll(FALSE))
return ResultFromScode(SELFREG_E_CLASS); return NOERROR; }; End File Listing:
AdsuraExchangeView.cpp Begin File Listing: AdsuraExchangeView.odl //
AdsuraExchangeView.odl : type library source for ActiveX Control project. // This file
will be processed by the Make Type Library (mktyplib) tool to // produce the type
library (AdsuraExchangeView.tlb) that will become a resource in //
AdsuraExchangeView.ocx. #include #include u[uid(B8189F11-CD0B-11D0-86E7-0000B43A75C9),
version(1.0), helpfile("AdsuraExchangeView.hlp"), helpstring("AdsuraExchangeView
ActiveX Control module"), control] library ADSURAEXCHANGEVIEWLib {
importlib(STDOLE_TLB); importlib(STDTYPE_TLB); // Primary dispatch interface for
CAdsuraExchangeViewCtrl u[uid(B8189F12-CD0B-11D0-86E7-0000B43A75C9),
helpstring("Dispatch interface for AdsuraExchangeView Control"), hidden ] dispinterface
_DAdsuraExchangeView { properties: // NOTE - ClassWizard will maintain property
information here. // Use extreme caution when editing this section. //
{{AFX_ODL_PROP(CAdsuraExchangeViewCtrl) [id(DISPID_READYSTATE), readonly] long
ReadyState; [id(1)] BSTR BaseValue; [id(2)] boolean DesignatedServer; [id(3)] BSTR
BaseDenom; //}}AFX_ODL_PROP methods: // NOTE - ClassWizard will maintain method
information here. // Use extreme caution when editing this section. // {{AFX_ODL_METHOD
(CAdsuraExchangeViewCtrl) //}}AFX_ODL_METHOD [id(DISPID_ABOUTBOX)] void AboutBox(); };
// Event dispatch interface for CAdsuraExchangeViewCtrl
u[uid(B8189F13-CD0B-11D0-86E7-0000B43A75C9), helpstring("Event interface for
AdsuraExchangeView Control") ] dispinterface DAdsuraExchangeViewEvents { properties: //
Event interface has no properties methods: // NOTE - ClassWizard will maintain event
information here. // Use extreme caution when editing this section. //
{{AFX_ODL_EVENT(CAdsuraExchangeViewCtrl) [id(DISPID_READYSTATECHANGE)] void
ReadyStateChange(); //}}AFX_ODL_EVENT }; // Class information for
CAdsuraExchangeViewCtrl u[uid(B8189F14-CD0B-11D0-86E7-0000B43A75C9),
helpstring("AdsuraExchangeView Control"), control ] coclass AdsuraExchangeView {
[default] dispinterface _DAdsuraExchangeView; [default, source] dispinterface
_DAdsuraExchangeViewEvents; }; // {{AFX_APPEND_ODL}} // }}AFX_APPEND_ODL}} }; End File
Listing: AdsuraExchangeView.odl Begin File Listing: AdsuraExchangeView.rc //Microsoft
Developer Studio generated resource script. // #include "resource.h" #define
APSTUDIO_READONLY_SYMBOLS //////////////////////////////////////////////////
////////////////////// // // Generated from the TEXTINCLUDE 2 resource. // #include
"afxres.h" //////////////////////////////////////////////
////////////////////// #undef APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////// ////////////////////////// //
English (Australia) resources #if !defined(AFX_RESOURCE_DLL) .vertline..vertline.
defined(AFX_TARG_ENA) #ifdef _WIN32 LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_AUS #pragma
```

```
code_page(1252) #endif // _WIN32 #ifdef APSTUDIO_INVOKED
////////////////////////////////////////////////////// ////////////////////////// // //
TEXTINCLUDE // 1 TEXTINCLUDE DISCARDABLE BEGIN "resource.h.backslash.0" END 2
TEXTINCLUDE DISCARDABLE BEGIN "#include ""afxres.h+".backslash.r.backslash.n"
".backslash.0" END 3 TEXTINCLUDE DISCARDABLE BEGIN "1 TYPELIB
""AdsuraExchangeView.tlb"".backslash.r.backslash.n" END #endif // APSTUDIO_INVOKED
#ifndef _MAC ////////////////////////////////////////////////////////
//////////////////////// // // Version // VS_VERSION_INFO VERSIONINFO FILEVERSION
1,0,0,1 PRODUCTVERSION 1,0,0,1 FILEFLAGSMASK 0x3fL #ifdef _DEBUG FILEFLAGS 0x1L #else
FILEFLAGS 0x0L #endif FILEOS 0x4L FILETYPE 0x2L FILESUBTYPE 0x0L BEGIN BLOCK
"StringFileInfo" BEGIN

Detailed Description Paragraph Table (15):
Begin File Listing: EBrowse.java
//********************************************************************************* ***** //
EBrowse.java: Applet //
//********************************************************************************* *****
import java.applet.*; import java.awt.*; import EBrowseFrame;
//================================================== ========================== //
Main Class for applet EBrowse // //==================================================
========================== public class EBrowse extends Applet { // STANDALONE
APPLICATION SUPPORT: // m_fStandAlone will be set to true if applet is run standalone
//-------------------------------------------------------------------- --------- private
boolean m_fStandAlone = true; // STANDALONE APPLICATION SUPPORT // The main( ) method
acts as the applet's entry point when it is run // as a standalone application. It is
ignored if the applet is run from // within an HTML page.
//-------------------------------------------------------------------- --------- public
static void main(String args[ ]) { // Create Toplevel Window to contain applet EBrowse
//-------------------------------------------------------------------- ---------
EBrowseFrame frame = new EBrowseFrame("EBrowse"); // Must show Frame before we size it
so insets( ) will return valid values
//-------------------------------------------------------------------- ---------
frame.show( ); frame.hide( ); // The following code starts the applet running within
the frame window. // It also calls GetParameters( ) to retrieve parameter values from
the // command line, and sets m_fStandAlone to true to prevent init( ) from // trying
to get them from the HTML page.
//-------------------------------------------------------------------- --------- EBrowse
applet_EBrowse = new EBrowse( ); frame.add("Center", applet_EBrowse);
applet_EBrowse.m_fStandAlone = true; applet_EBrowse.init( ); applet_EBrowse.start( );
frame.show( ); } // EBrowse Class Constructor
//-------------------------------------------------------------------- --------- public
EBrowse( ) { // TODO: Add constructor code here } // APPLET INFO SUPPORT: // The
getAppletInfo( ) method returns a string describing the applet's // author, copyright
date, or miscellaneous information.
//-------------------------------------------------------------------- ----- public
String getAppletInfo( ) { return "Name: EBrowse.backslash.r.backslash.n" + "Author:
Mark Wickman.backslash.r.backslash.n" + "Created with Microsoft Visual J++ Version
1.1"; } // The init( ) method is called by the AWT when an applet is first loaded or //
reloaded. Override this method to perform whatever initialization your // applet needs,
such as initializing data structures, loading images or // fonts, creating frame
windows, setting the layout manager, or adding UI // components.
//-------------------------------------------------------------------- ----- public
void init( ) { } // Place additional applet clean up code here. destroy( ) is called
when // when you applet is terminating and being unloaded.
//-------------------------------------------------------------------- ----- public
void destroy( ) { // TODO: Place applet cleanup code here } // EBrowse Paint Handler
//-------------------------------------------------------------------- ----- public
void paint(Graphics g) { g.drawString("Adsura Exchange Browser Sample, Please run as a
standalone application", 10, 20); } // The start( ) method is called when the page
containing the applet // first appears on the screen. The AppletWizard's initial
implementation // of this method starts execution of the applet's thread.
//-------------------------------------------------------------------- --------- public
void start( ) { // TODO: Place additional applet start code here } // The stop( )
method is called when the page containing the applet is // no longer on the screen. The
AppletWizard's initial implementation of // this method stops execution of the applet's
thread. //-------------------------------------------------------------------- ---------
public void stop( ) { } } End File Listing: EBrowse.java Begin File Listing:
EBrowseFrame.java
//********************************************************************************* ***** //
```

```
EBrowseFrame.java:
//********************************************************************** *****
import java.awt.*; import java.net.*; import java.io.*; import java.util.*; //Test
Address at: // http://sabre/testweb/parse.htm
//================================================ ======================== //
STANDALONE APPLICATION SUPPORT // This frame class acts as a top-level window in which
the applet appears // when it's run as a standalone application.
//================================================ ======================== class
EBrowseFrame extends Frame { Vector m_vConversionRates = new Vector(20); Vector
m_vCurrencyNames = new Vector(20); boolean m_bInTag; String m_strAddress; void
mCCurrencySel_Action(Event event) { mCAddress_EnterHit(null); } void
mCAddress_EnterHit(Event event) { // Get the file pointed to by the text field and
retrieve it locally then crunch the page m_strAddress = m_cAddress.getText( ); try {
URL urlAddress = new URL(m_strAddress); DataInputStream disInput = new
DataInputStream(urlAddress.openStream( )); m_cDisplayArea.setText(" "); String
strInputline; m_bInTag = false; while( (strInputline = disInput.readLine( )) != null) {
//System.out.println(strInputline); ParseLine(strInputline); } disInput.close( ); }
catch(MalformedURLException e) { m_cAddress.setText(" ");
System.out.println("MalformedURLException: " + e); return; } catch(IOException e) {
m_cAddress.setText(" "); System.out.println("IOException: " + e); return; } } void
ParseLine(String strInput) { String strStripped = new String(StripTags(strInput));
if(strStripped.length( ) > 0) { m_cDisplayArea.appendText(strStripped);
m_cDisplayArea.appendText(".backslash.r.backslash.n"); } } String StripTags(String
strRaw) { String strTagValue = new String( ); String strTagContent = new String( );
for(int i = 0; i < strRaw.length( ); i++) { char cTest = strRaw.charAt(i);
switch(cTest) { case `<`: m_bInTag = true; break; case `>`: m_bInTag = false; break;
default: if(!m_bInTag) strTagValue += cTest; else strTagContent += cTest; break; } }
return(ProcessTag(strTagContent, strTagValue)); } String ProcessTag(String strTag,
String strTagValue) { if(strTag.equalsIgnoreCase("title/title") == true) {
setTitle(strTagValue); return(" "); } int nTemp = -1; if( (nTemp =
strTag.indexOf("act")) != -1 && strTag.indexOf("act",nTemp + 1) != -1) { //Adsura
Couplet Technology Tag //Strip out the Denominator and Amount int nDenom =
strTag.indexOf("denom"); int nDenomStart = strTag.indexOf(".backslash.",nDenom + 4);
int nDenomEnd = strTag.indexOf(".backslash.",nDenomStart + 1); String strDenom = new
String(strTag.substring(nDenomStart + 1,nDenomEnd)); int nAmount =
strTag.indexOf("amount"); int nAmountStart = strTag.indexOf(".backslash.", nAmount +
4); int nAmountEnd = strTag.indexOf(".backslash.", nAmountStart + 1); String strAmount
= new String(strTag.substring(nAmountStart + 1,nAmountEnd));
return(CalculateDisplayAmount(strDenom,strAmount)); } return(strTagValue); } String
CalculateDisplayAmount(String strDenom, String strAmount) { String strChosen = new
String((String)m_vCurrencyNames.elementAt(m_CurrencySel.g etSelectedIndex( )));
if(strDenom.compareTo(strChosen) == 0) return(strChosen + " " + strAmount); //OK, it is
not the default //Now get the Value from the DataFile Vector that Matches the String in
the Currency Unit Vector Enumeration e = m_vCurrencyNames.elements( ); int nCount = 0;
while(e.hasMoreElements( )) { String strTemp = new String((String) e.nextElement( ));
if(strTemp.compareTo(strDenom) == 0) break; nCount++; } float fEmbeddedRate =
((Float)m_vConversionRates.elementAt(nCount)).floatValue( ); float fDefaultAmount = (
(new Float(strAmount)).floatValue( )); float fSelectedRate =
((Float)m_vConversionRates.elementAt(m_cCurrencySel.getSel ectedIndex( ))).floatValue(
); //To calculate the Exchanger Rate Ratio (Selected Rate/Default (Embedded) Rate)
float fRatio = (float) fSelectedRate/fEmbeddedRate; //To Calculate the converted Value
multiply the Rate Ratio by the Original Value Float fConverted = new
Float(fDefaultAmount * fRatio); Integer nRounded = new Integer((int)
Math.floor(fConverted.floatValue( ))); Float fRemainder = new Float((float)
(fConverted.floatValue( )) % nRounded.intValue( )); String strRounded = new
String(nRounded.toString( )); String strTemp = new String(fRemainder.toString( ));
String strHold = new String(" "); int nCharCount = 0; boolean bStartAdding = false;
for(int i = 0; i < strTemp.length( ); i++) { char cTest = strTemp.charAt(i);
switch(cTest) { case `.`: bStartAdding = true; break; default: if(bStartAdding &&
nCharCount < 2)
```

CLAIMS:

11. A method for performing currency conversion utilizing a computer coupled to a network, comprising the steps of:

(a) displaying one or more prices utilizing logic for establishing the default amount and currency by interpreting parameters embedded in a document or environment;

(b) displaying the price in the appropriate currency by reading operating system information indicative of a desired currency;

(c) retrieving an exchange rate information required to perform a conversion to the desired currency by network query; and

(d) allowing use of a display to select the desired currency during the display of one or more of the prices on the display.

21. A computer program embodied on a computer-readable medium for performing currency conversion utilizing a computer coupled to a network, comprising:

(a) a code segment that displays one or more prices utilizing logic for establishing a default currency which interprets parameters embedded in a document or environment;

(b) a code segment that displays the price in the appropriate currency by reading operating system variables indicative of a desired currency;

(c) a code segment that retrieves exchange rate information to convert a desired currency utilizing the network to retrieve the exchange rate information; and

(d) a code segment that allows use of a display to select the desired currency during the display of one or more of the prices on the display.